**Please note that Cypress is an Infineon Technologies Company.**
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

**Continuity of document content**
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

**Continuity of ordering part numbers**
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

# CE219656 - PSoC 6 MCU UART using Low Level APIs

## Objective

This example demonstrates the UART transmit and receive operation in PSoC® 6 MCU using low level APIs. This is done using polling, ISR, and DMA methods.

## Overview

This example contains three sub-examples that implement a UART using: polling, ISR, or DMA to manage the UART. Each sub-example uses low level UART APIs and echoes what is received on the UART serial terminal. The UART_Low_Level_Polling example polls repeatedly. The UART_Low_Level_User_ISR example uses a user interrupt. The UART_Low_Level_DMA example uses DMA functions.

## Requirements

**Tool:** PSoC Creator™ 4.2

**Programming Language:** C (ARM® GCC 5.4-2016-q2-update, ARM MDK 5.22)

**Associated Parts:** All PSoC 6 MCU parts

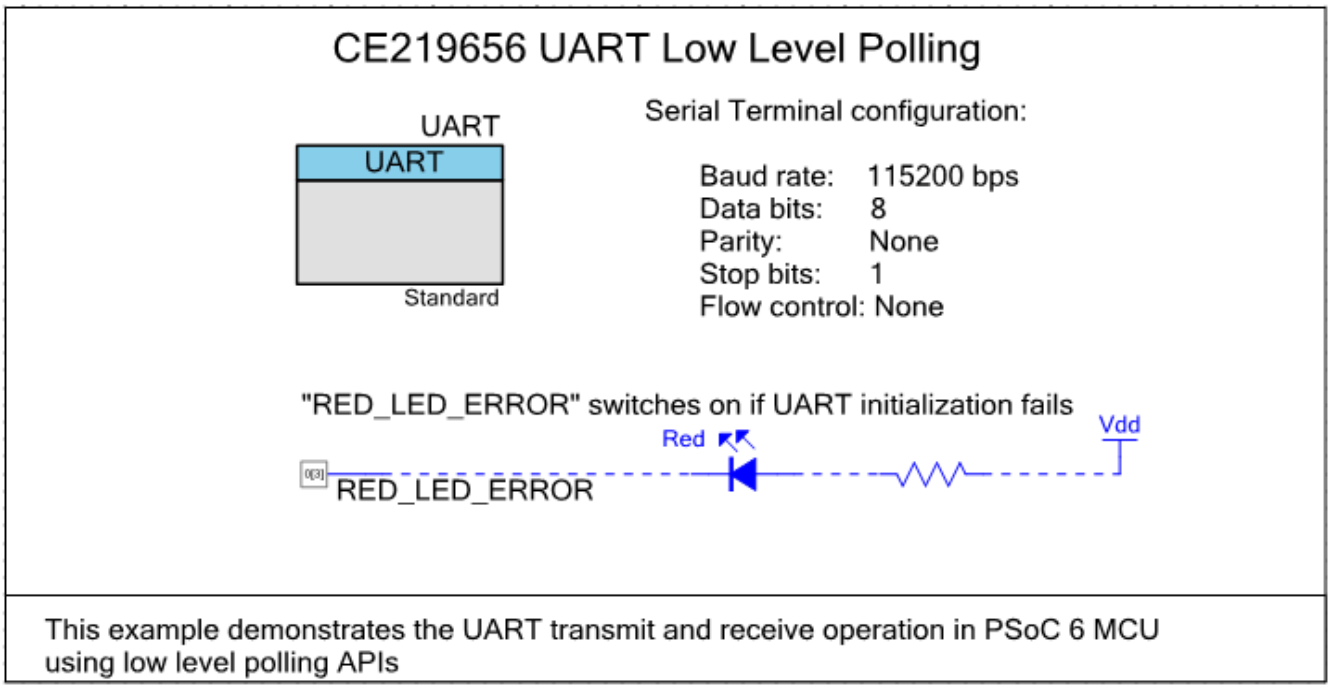
**Related Hardware:** CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit

# Design

## UART_Low_Level_Polling

The UART_Low_Level_Polling design shown in Figure 1 has a UART (SCB_UART_PDL) Component configured for TX+RX mode at 115200 bps baud rate, 8N1.

Figure 1. The UART_Low_Level_Polling Example Schematic



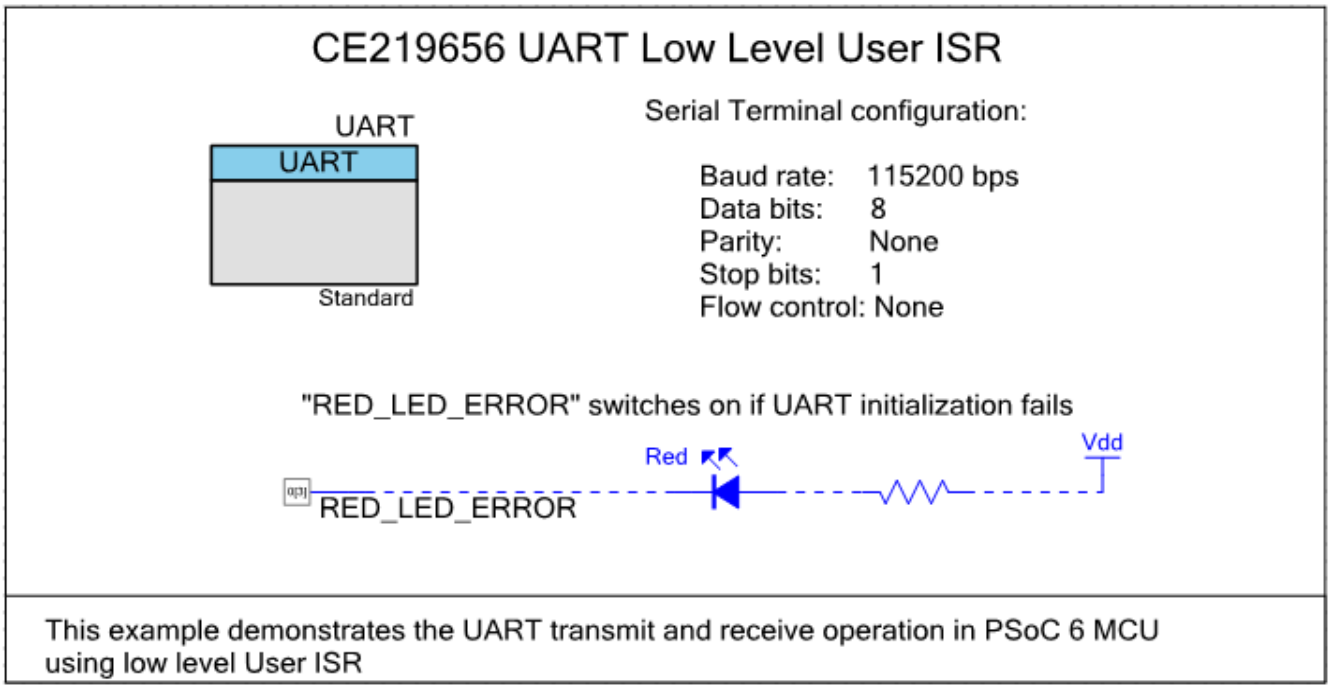

The UART_Low_Level_Polling example firmware performs the following functions:

1. Turns OFF the RED_LED_ERROR.
2. Configures the UART Component.
3. Turns ON the RED_LED_ERROR if the UART Component initialization fails, then stays in an infinite loop.
4. Uses the UART to send a text header to the serial terminal.
5. Uses the low-level UART API to retransmit whatever the user types on the console.

## UART_Low_Level_User_ISR

Figure 2 shows the UART_Low_Level_User_ISR design. This design has a UART (SCB_UART_PDL) Component configured for TX+RX mode at 115200 bps baud rate, 8N1, and Interrupt mode: Internal.

Figure 2. The UART_Low_Level_User_ISR Example Schematic




The UART_Low_Level_User_ISR example firmware performs the following functions in main:

1. Turns OFF the RED_LED_ERROR.
2. Configures the UART Component.
3. Turns ON the RED_LED_ERROR if the UART Component initialization fails, then stays in an infinite loop.
4. Uses the UART to send a text header to the serial terminal.
5. Configures the UART interrupt with ISR_UART as interrupt handler and enables UART interrupt.
6. Waits in an infinite loop for “UART RX FIFO not empty” interrupt.

The UART_Low_Level_User_ISR example firmware performs the following functions in ISR_UART:

1. If the interrupt cause is “UART RX FIFO not empty”, it clears the interrupt, gets the received character from the terminal, and echoes the character to the terminal.
2. If the interrupt cause is not “UART RX FIFO not empty”, it turns ON the RED_LED_ERROR and stays in an infinite loop.

## UART_Low_Level_DMA

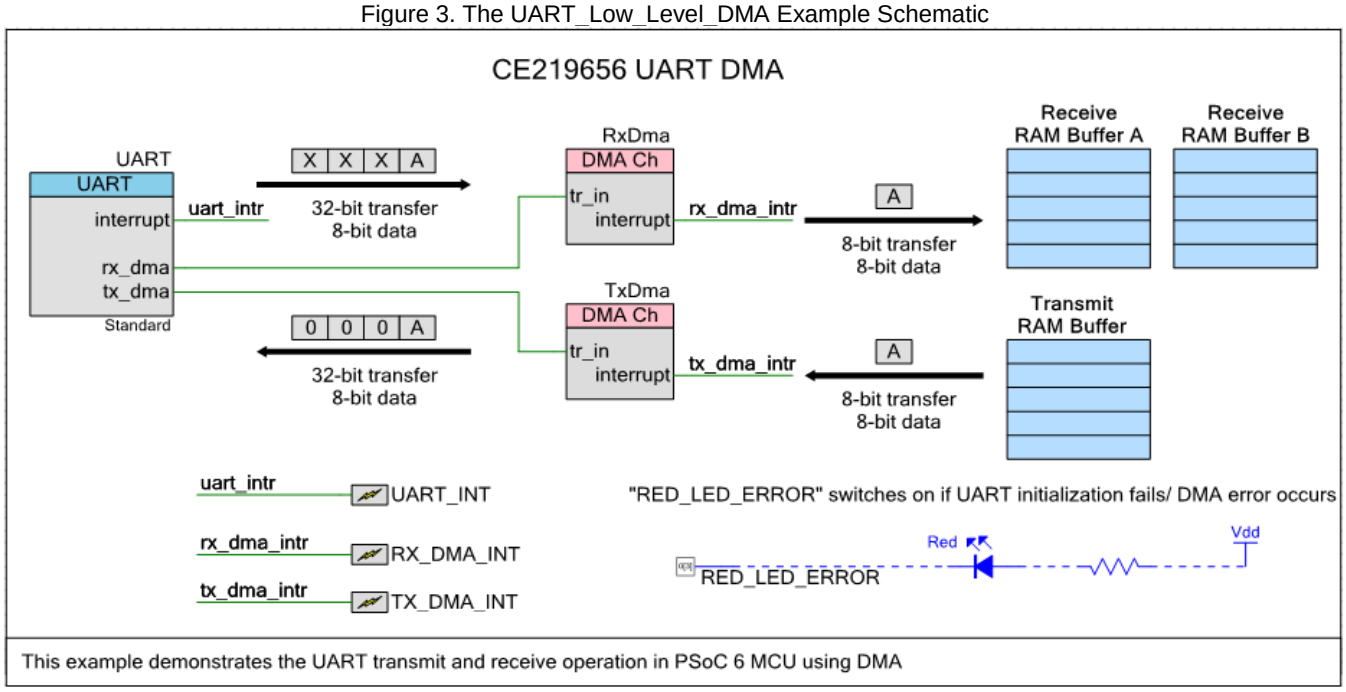
The UART_Low_Level_DMA design shown in Figure 3 has a UART (SCB_UART_PDL) Component configured for TX+RX mode at 115200 bps baud rate, 8N1.

The rx_dma and tx_dma interrupt outputs are enabled in the Component configuration. The RxDma Component handles data transfer in the receive direction. The RxDma has two descriptors in the chain. These two descriptors are configured such that the source buffer becomes ping pong in the receive direction to provide firmware time to pull the data out of one or the other buffer. Each descriptor is configured to transfer a single element in byte mode. The TxDma Component handles data transfer in the transmit direction. The TxDma has only one descriptor, configured to transfer a single element in byte mode.

The UART_INT (System Interrupt) Component is configured to handle UART error interrupts - RX FIFO Overflow, RX FIFO Underflow, and TX FIFO Overflow. RX_DMA_INT (System Interrupt) Component is configured to handle UART rx_dma interrupt. TX_DMA_INT (System Interrupt) Component is configured to handle UART tx_dma interrupt.

The RED_LED_ERROR indicates UART initialization failure status or DMA error status.

Figure 3. The UART_Low_Level_DMA Example Schematic




The UART_Low_Level_DMA example firmware performs the following functions in main:

1. Configures RxDma:
    a. Initializes the RxDma_Descriptor_1 with source address as UART RX FIFO and destination address as RxDmaUartBufferA SRAM buffer
    b. Initializes the RxDma_Descriptor_2 with source address as UART RX FIFO and destination address as RxDmaUartBufferB SRAM buffer
    c. Initializes the rx_dma interrupt with RxDmaComplete as interrupt handler and enables rx_dma interrupt
    d. Enables the RxDma channel
2. Configures TxDma:
    a. Initializes the TxDma_Descriptor_1 with source address as RxDmaUartBufferA SRAM buffer and destination address as UART TX FIFO. Note that the TxDma_Descriptor_1 source address ping pongs between RxDmaUartBufferA and RxDmaUartBufferB during data transfer
    b. Initializes the tx_dma interrupt with TxDmaComplete as interrupt handler and enables tx_dma interrupt
    c. Enables the TxDma channel

3. Initializes and enables the UART error interrupts
4. Switch OFF the RED_LED_ERROR
5. Configures the UART Component
6. Switches on the RED_LED_ERROR if the UART Component initialization fails and stays in an infinite loop
7. Uses the UART to send a text header into the serial terminal
8. Enables the global interrupts
9. Waits in an infinite loop for any of the rx_dma interrupt, tx_dma interrupt, or UART error interrupt.

### Design Considerations

This code example is designed to run on CY8CKIT-062-BLE with the PSoC 6 MCU device. To port the design to other PSoC 6 MCU family devices and kits, you must change the target device in Device Selector, and change the pin assignments in the ***cydwr*** settings. For single-core PSoC 6 MCU devices, port the code from *main_cm4.c* to *main.c* file as CM0+ CPU is not used in this code example.

## Hardware Setup

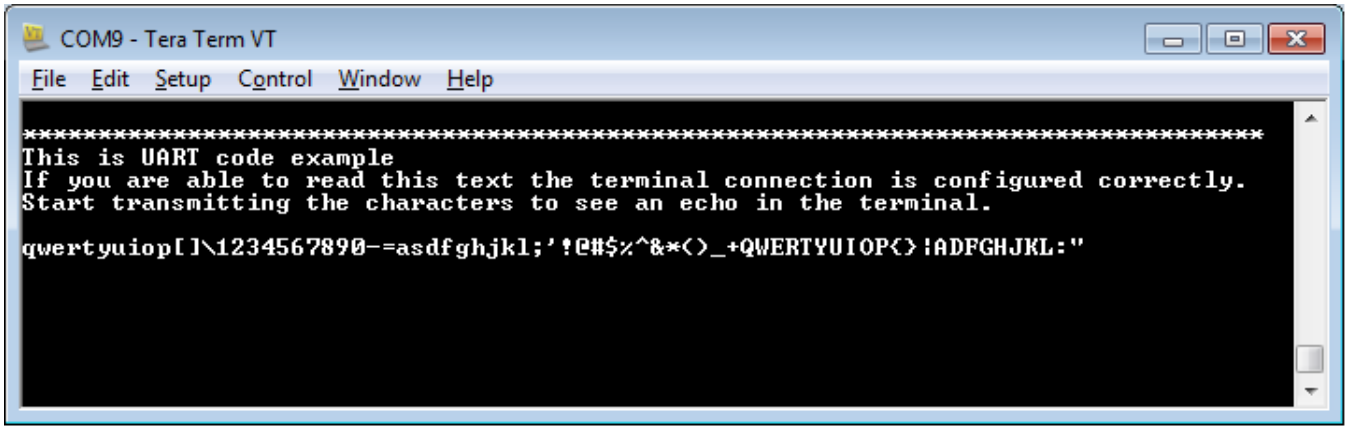
The code example works with the default settings on the CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit. If the settings are different from the default values, see the “Selection Switches” table in the kit guide to reset to the default settings.

## Operation

1. Connect the CY8CKIT-062-BLE Kit or CY8CKIT-062 Kit to a USB port on your PC.
2. Open a serial port communication program such as Tera Term and select the corresponding COM port. Configure the terminal to match the UART: 115200 baud rate, 8N1, and Flow control – None. These settings must match the configuration of the PSoC Creator UART Component in the project.
3. Build and program the application into the CY8CKIT-062-BLE Kit or CY8CKIT-062 Kit. For more information on building a project or programming a device, see *PSoC Creator Help*.
4. Observe the UART example header message printed in the terminal window.
5. Type any character in the terminal window, and make sure that the same character is displayed in the terminal.

Figure 4 shows a snapshot of a sample UART terminal output.

Figure 4. UART Terminal Output




## Components

Table 1 lists the PSoC Creator Components used in all three sub-examples and the hardware resources used by each Component.

Table 1. PSoC Creator Components

| Component | Instance Name | Hardware Resources |
|---|---|---|
| UART (SCB_UART_PDL) | UART | Single SCB peripheral block |
| General Purpose Input / Output (GPIO) | RED_LED_ERROR | One physical pin |
| Direct Memory Access (DMA_PDL) | RxDma, TxDma | Two DMA channels |
| System Interrupt (SysInt) | UART_INT, RX_DMA_INT, TX_DMA_INT | Three entries in the device interrupt vector table |

## Parameter Settings

Non-default settings for each Component are outlined in red in the following figures.

Figure 5 shows the UART_Low_Level_DMA example UART Component parameter settings.

Figure 5. UART_Low_Level_DMA Example UART Component Parameter Settings

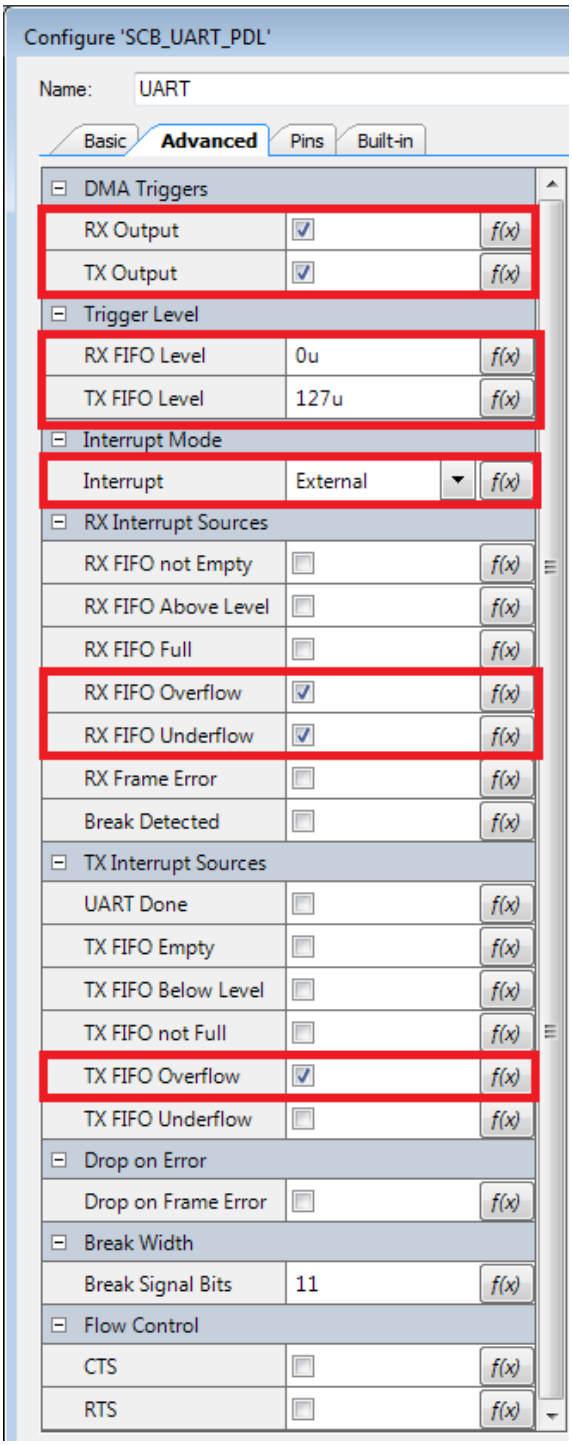


Figure 6 shows the UART_Low_Level_DMA example RxDma Component parameter settings.

Figure 6. UART_Low_Level_DMA Example RxDma Component Parameter Settings

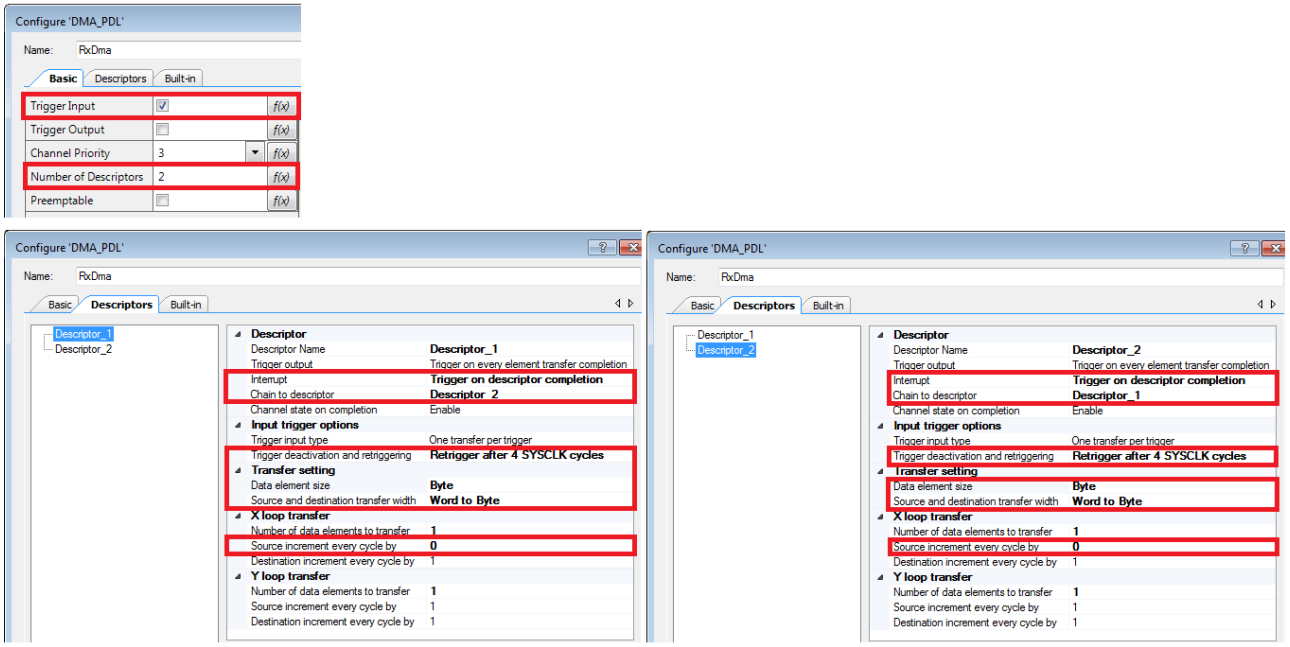
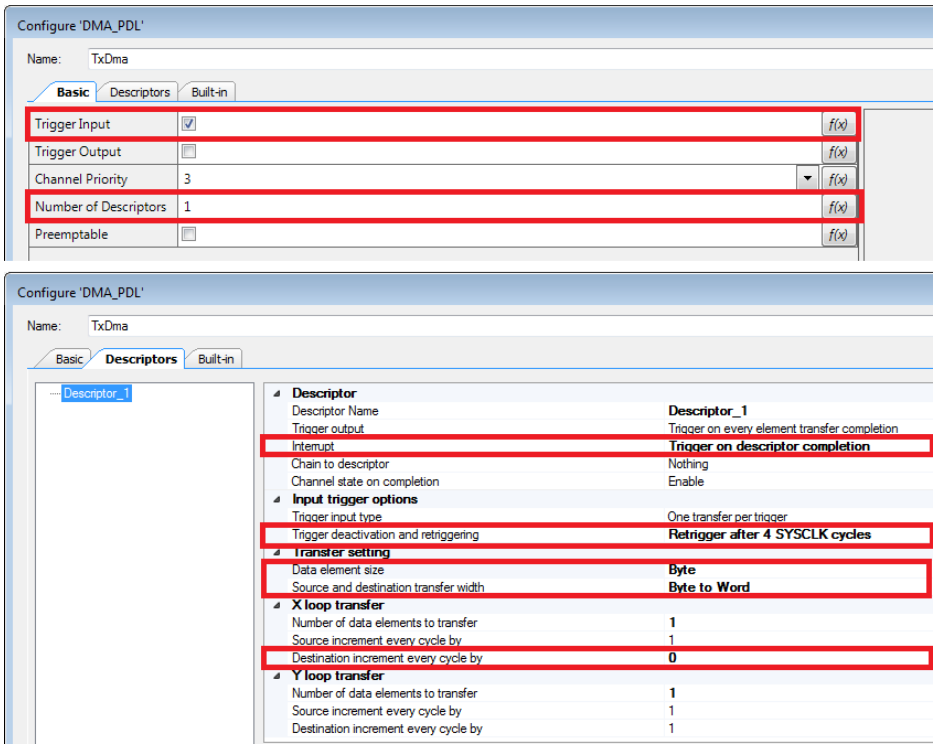



Figure 7 shows the UART_Low_Level_DMA example TxDma Component parameter settings.

Figure 7. UART_Low_Level_DMA Example TxDma Component Parameter Settings



### Design-Wide Resources

Table 2 shows the pin assignment for the code example.

Table 2. Pin Names and Location

| Pin Name | Location |
|---|---|
| UART:rx | P5[0] |
| UART:tx | P5[1] |
| RED_LED_ERROR | P0[3] |

## Related Documents

Table 3 lists all relevant application notes, code examples, knowledge base articles, device datasheets, and Component/user module datasheets.

Table 3. Related Documents

| | |
|---|---|
| **Application Notes** | |
| AN210781 Getting Started with PSoC 6 MCU with Bluetooth Low Energy (BLE) Connectivity | Describes PSoC 63 with Bluetooth Low Energy (BLE) Connectivity and how to build your first PSoC Creator project |
| **PSoC Creator Component Datasheets** | |
| UART | Supports UART communication |
| Direct Memory Access | Supports up to 16 DMA channels |
| System Interrupt | Interrupt vectoring and control |
| General-Purpose Input / Output | Supports Analog, Digital I/O and Bidirectional signal types |
| **Device Documentation** | |
| PSoC 6 MCU: PSoC 63 with BLE Datasheet | PSoC 6 MCU: PSoC 63 with BLE Architecture Technical Reference Manual |
| **Development Kit (DVK) Documentation** | |
| CY8CKIT-062-BLE PSoC 6 BLE Pioneer Kit | |

## Document History

Document Title: CE219656 - PSoC 6 MCU UART using Low Level APIs

Document Number: 002-19656

| Revision | ECN | Orig. of Change | Submission Date | Description of Change |
|---|---|---|---|---|
| *A | 5856608 | VJYA | 08/23/2017 | Initial public release |
| *B | 6003214 | VJYA | 12/22/2017 | Updated to latest PSoC Creator build and Fixed CDT#295974 |

## Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

## Products

| | |
|---|---|
| Arm® Cortex® Microcontrollers | cypress.com/arm |
| Automotive | cypress.com/automotive |
| Clocks & Buffers | cypress.com/clocks |
| Interface | cypress.com/interface |
| Internet of Things | cypress.com/iot |
| Memory | cypress.com/memory |
| Microcontrollers | cypress.com/mcu |
| PSoC | cypress.com/psoc |
| Power Management ICs | cypress.com/pmic |
| Touch Sensing | cypress.com/touch |
| USB Controllers | cypress.com/usb |
| Wireless Connectivity | cypress.com/wireless |

## PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6 MCU

## Cypress Developer Community

Forums | WICED IOT Forums | Projects | Videos | Blogs | Training | Components

## Technical Support

cypress.com/support

All other trademarks or registered trademarks referenced herein are the property of their respective owners.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

© Cypress Semiconductor Corporation, 2017. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. No computing device can be absolutely secure. Therefore, despite security measures implemented in Cypress hardware or software products, Cypress does not assume any liability arising out of any security breach, such as unauthorized access to or use of a Cypress product. In addition, the products described in these materials may contain design defects or errors known as errata which may cause the product to deviate from published specifications. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.